An Abstract Interpretation Perspective on Linear vs. Branching Time[⋆]

Francesco Ranzato and Francesco Tapparo

Dipartimento di Matematica Pura ed Applicata,
Università di Padova, Italy

Abstract. It is known that the branching time language ACTL and the linear time language $\forall\mathrm{LTL}$ of universally quantified formulae of LTL have incomparable expressive powers, i.e., $\mathrm{Sem}(\mathrm{ACTL})$ and $\mathrm{Sem}(\forall\mathrm{LTL})$ are incomparable sets. Within a standard abstract interpretation framework, ACTL can be viewed as an abstract interpretation $\mathrm{LTL}^{\forall}$ of LTL where the universal path quantifier $\forall$ abstracts each linear temporal operator of LTL to a corresponding branching state temporal operator of ACTL. In abstract interpretation terms, it turns out that the universal path quantifier abstraction of LTL is incomplete. In this paper we reason on a generic abstraction α over a domain A of a generic linear time language L. This approach induces both a language $\alpha\mathrm{L}$ of α-abstracted formulae of L and an abstract language L^{α} whose operators are the best correct abstractions in A of the linear operators of L. When the abstraction α is complete for the operators in L it turns out that $\alpha\mathrm{L}$ and L^{α} have the same expressive power, so that trace-based model checking of $\alpha\mathrm{L}$ can be reduced with no lack of precision to A-based model checking of L^{α}. This abstract interpretation-based approach allows to compare temporal languages at different levels of abstraction and to view the standard linear vs. branching time comparison as a particular instance.

1 Introduction

The relationship between linear and branching time specification languages to be used in automatic system verification by model checking has been the subject of thorough investigation [2,8,11,12,13,14,19] (see [20] for a survey). In particular, some of these works [2,8,11,13,14] studied the relationship between the expressive power of linear vs. branching time formalisms.

LTL and CTL are the most commonly used languages for, respectively, linear and branching time model checking. ACTL is the fragment of CTL that uses only the universal path quantifier. Given a Kripke structure $\mathcal{K} = (\Sigma, \xrightarrow{R})$, the standard approach for comparing a linear formula $\varphi \in \mathrm{LTL}$ and a branching formula $\psi \in \mathrm{CTL}$ consists in "abstracting" the path semantics $[\![\varphi]\!] = \{\pi \in \mathit{Path}(\mathcal{K}) \mid \pi \models \varphi\}$ to its corresponding universal (or, dually, existential) state semantics $\{s \in \Sigma \mid \forall \pi \in \mathit{Path}(\mathcal{K}).\, (\pi(0) = s) \Rightarrow \pi \models \varphi\}$ and then comparing this set of states with the standard state semantics $[\![\psi]\!] = \{s \in \Sigma \mid s \models \psi\}$ of ψ. As shown by Cousot and Cousot [5], the intuition

⋆ This work was partially supported by the FIRB Project "Abstract interpretation and model checking for the verification of embedded systems" and by the COFIN2004 Project "AIDA".

that this actually is a step of abstraction can be precisely formalized within the abstract interpretation framework [3,4]. In fact, Cousot and Cousot [5] show that the universal path quantifier is an abstraction function $\forall : \wp(\mathrm{Trace}(\Sigma)) \rightarrow \wp(\Sigma)$ mapping any set T of traces, viz. arbitrary sequences of states, to the set of states $s \in \Sigma$ such that any path in $\mathcal{K}$ that begins in s belongs to T.

The standard approach introduced by Emerson and Halpern [8] for comparing linear and universal branching time languages relies on the above universal branching abstraction $\forall$. If $\mathrm{L} \subseteq \mathrm{LTL}$ is a linear time language and $\mathcal{L} \subseteq \mathrm{ACTL}$ is a branching time language then L and $\mathcal{L}$ can be compared by comparing the sets $\{\forall(\llbracket \varphi \rrbracket) \subseteq \Sigma \mid \varphi \in \mathrm{L}\}$ and $\{\llbracket \psi \rrbracket \subseteq \Sigma \mid \psi \in \mathcal{L}\}$ in $\wp(\wp(\Sigma))$. Thus, the linear time language L is abstracted to a universal branching time language $\forall\mathrm{L}$, denoted by $\mathrm{B(L)}$ in [8]. For example, it is well known that LTL and ACTL are incomparable (cf. [2,8]), where this means that $\forall\mathrm{LTL}$ and ACTL are incomparable in $\wp(\wp(\Sigma))$.

Moreover, if L is a linear time language which is inductively generated by a set of linear operators $f \in Op_{\mathrm{L}}$ (and a set of atomic propositions p), i.e., $\mathrm{L} \ni \varphi ::= p \mid f(\varphi_1, ..., \varphi_n)$, then the universal path quantifier also induces the following universal state language: $\mathrm{L}^{\forall} \ni \psi ::= \forall p \mid \forall f(\psi_1, ..., \psi_n)$, where each linear temporal operator in Op is preceded by the universal path quantifier $\forall$. For example, it turns out that $\mathrm{ACTL} = \mathrm{LTL}^{\forall}$. Thus, the comparison between $\forall\mathrm{LTL}$ and ACTL boils down to the comparison between $\forall\mathrm{LTL}$ and $\mathrm{LTL}^{\forall}$. As a consequence of the incomparability of $\forall\mathrm{LTL}$ and $\mathrm{LTL}^{\forall}$ we obtain that the abstraction map $\forall$ is *incomplete* in the abstract interpretation sense [3,9]. In fact, if $\forall$ would be complete for the operators of LTL then we would also have that $\forall\mathrm{LTL} = \mathrm{LTL}^{\forall}$ whereas this is not the case. Cousot and Cousot [5] analyzed the linear operators that cause the incompleteness of $\forall$ and then isolated some inductive fragments $\mathrm{L} \subseteq \mathrm{LTL}$ such that $\forall\mathrm{L} = \mathrm{L}^{\forall}$.

Thus, abstract interpretation allows to cast the linear vs. branching time problem as a particular instance of a more general "linear vs. A time" problem, where $\alpha : \wp(\mathrm{Trace}(\Sigma)) \rightarrow A$ is any abstract interpretation of sets of traces to some abstract domain A. For any such abstraction α, a linear language L therefore induces two "A-time" languages: $\alpha\mathrm{L}$ and L^{α}.

In this paper, we study a number of abstractions of sets of traces that are alternative to the above standard universal path quantifier abstraction. We consider abstractions of $\wp(\mathrm{Trace}(\Sigma))$ parameterized by some model M, namely by the set $Path(\mathcal{K})$ of paths in some Kripke structure $\mathcal{K}$. This is more general than considering abstractions of $\wp(Path(\mathcal{K}))$ because we show that $\wp(Path(\mathcal{K}))$ is a complete (both existential and universal) abstract interpretation of $\wp(\mathrm{Trace}(\Sigma))$. Completeness plays a key role in this generalized approach. In fact, it turns out that when α is complete for the linear operators in Op_{L} of some language L then $\alpha\mathrm{L} = \mathrm{L}^{\alpha}$. We first study an abstract domain consisting of *traces of sets of states*, i.e., $\mathrm{Trace}(\wp(\Sigma))$. Here, the trace of sets abstraction $\mathrm{tr} : \wp(\mathrm{Trace}(\Sigma)) \rightarrow \mathrm{Trace}(\wp(\Sigma))$ approximates any set T of traces to the sequence of sets of states reached by some trace in T. This is a more precise abstraction than the universal branching abstraction $\forall$. While tr is not complete for all the linear operators of LTL, we show that tr is instead complete for disjunction, next and eventually operators, namely for the fragment $\mathrm{L}(\{\vee, \mathrm{X}, \mathrm{F}\}) \subseteq \mathrm{LTL}$. We then consider a *reachable state* abstraction $\mathrm{rs} : \wp(\mathrm{Trace}(\Sigma)) \rightarrow \wp(\Sigma)$, where any set T of traces T is approximated to the set of states reached by some trace in T. This abstraction is incomparable with the universal branching abstraction $\forall$ while it is less precise than the trace of sets abstraction. In this case, we show that rs is not complete for the next operator and it is still complete for the fragment $\mathrm{L}(\{\vee, \mathrm{F}\})$.

This abstract interpretation-based perspective of the linear vs. branching time problem allows us to show that the Emerson and Halpern [8] transform $\mathrm{EH}_\forall$ of a linear time language L to the branching time language $\forall\mathrm{L}$ actually can be viewed as a "higher-order" abstract interpretation. This means that $\mathrm{EH}_\forall$ is an abstraction from trace abstract domains to universal branching state abstract domains. Hence, the Emerson and Halpern transform can be generalized to a higher-order abstract interpretation $\mathcal{A}_\alpha : \mathrm{AbsDom}(\wp(\mathrm{Trace}(\Sigma))) \to \mathrm{AbsDom}(A)$ which is parameterized by any trace abstraction $\alpha : \wp(\mathrm{Trace}(\Sigma)) \to A$, so that $\mathcal{A}_\alpha(\mathrm{L}) = \{\alpha([\![\varphi]\!]) \mid \varphi \in \mathrm{L}\}$. Therefore, this generalized approach allows to compare the expressive power of linear time languages at any level of abstraction A. As an example, we consider the linear time language $\mathrm{L} \ni \varphi ::= p \mid \mathrm{F}\varphi \mid \mathrm{G}\varphi$. We show how this approach can be used to prove that the languages $\forall\mathrm{L}$ and $\mathrm{L}^\forall$ have incomparable expressive powers by comparing them in a higher-order abstract domain of state partitions.

2 Basic Notions

Notation. Let X be any set. When writing a set $S \in \wp(\wp(X))$ we often use a compact form like in $\{1, 12, 123\} \in \wp(\wp(\{1, 2, 3\}))$. We denote by $\neg$ the complement operator w.r.t. some universe set. A poset or complete lattice C w.r.t. a partial ordering $\leq$ is denoted by $C_\leq$ or $\langle C, \leq \rangle$. A function $f : C \to C$ on a complete lattice C is additive when f preserves arbitrary least upper bounds. We denote by $\mathrm{Part}(X)$ the set of partitions of X. $\mathrm{Part}(X)$ is endowed with the following standard partial order $\preccurlyeq$: given $P_1, P_2 \in \mathrm{Part}(X)$, $P_1 \preccurlyeq P_2$ (P_1 refines P_2) iff $\forall B \in P_1 . \exists B' \in P_2 . B \subseteq B'$. It turns out that $\langle \mathrm{Part}(X), \preccurlyeq \rangle$ is a complete lattice.

Kripke Structures. We consider transition systems (Σ, R) where the transition relation $R \subseteq \Sigma \times \Sigma$ (also denoted by $\xrightarrow{R}$) is total. A Kripke structure $\mathcal{K} = (\Sigma, R, AP, \ell)$ consists of a transition system (Σ, R) together with a set AP of atomic propositions and a labeling function $\ell : \Sigma \to \wp(AP)$. A trace on Σ is any infinite sequence of elements in Σ, that is, any function $\sigma : \mathbb{N} \to \Sigma$. $\mathrm{Trace}(\Sigma)$ denotes the set of traces on Σ. For any $k \in \mathbb{N}$ and $\sigma \in \mathrm{Trace}(\Sigma)$, $\sigma^k \in \mathrm{Trace}(\Sigma)$ denotes the suffix of σ that begins in $\sigma(k)$, i.e., $\sigma^k = \lambda i \in \mathbb{N}. \sigma(i + k)$. A path in a Kripke structure $\mathcal{K}$ (or, more in general, in a transition system) is any trace $\pi \in \mathrm{Trace}(\Sigma)$ such that for any $i \in \mathbb{N}$, $\pi(i) \xrightarrow{R} \pi(i + 1)$. $\mathit{Path}(\mathcal{K})$ denotes the set of paths in $\mathcal{K}$.

Temporal Languages. LTL and ACTL are two well-known temporal specification languages used in model checking. LTL consists of linear (or path) formulae describing properties of a computation path through linear temporal operators. ACTL consists of branching (or state) formulae that describe properties of computation trees because each temporal operator is preceded by the universal path quantifier. We consider formulae in negation-normal form, so that LTL is inductively defined as follows:

$$\mathrm{LTL} \ni \varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \mathrm{X}\varphi \mid \mathrm{U}(\varphi_1, \varphi_2) \mid \mathrm{V}(\varphi_1, \varphi_2)$$

where p ranges over a set AP of atomic propositions that contains $true$. Given a Kripke structure $\mathcal{K}$, let us recall the standard semantics $[\![\cdot]\!]_{\mathcal{K}} : \mathrm{LTL} \rightarrow \wp(Path(\mathcal{K}))$ of LTL:

- $[\![p]\!]_{\mathcal{K}} = \{\pi \in Path(\mathcal{K}) \mid p \in \ell(\pi(0))\}$;
- $[\![\neg p]\!]_{\mathcal{K}} = \mathrm{Path}(\mathcal{K}) \smallsetminus [\![p]\!]_{\mathcal{K}}$;
- $[\![\varphi_1 \wedge/\vee\ \varphi_2]\!]_{\mathcal{K}} = [\![\varphi_1]\!]_{\mathcal{K}} \cap/\cup\ [\![\varphi_2]\!]_{\mathcal{K}}$;
- "Next": $[\![\mathrm{X}\varphi]\!]_{\mathcal{K}} = \{\pi \in Path(\mathcal{K}) \mid \pi^1 \in [\![\varphi]\!]_{\mathcal{K}}\}$;
- "Until": $[\![\mathrm{U}(\varphi_1, \varphi_2)]\!]_{\mathcal{K}} = \{\pi \in Path(\mathcal{K}) \mid \exists k \in \mathbb{N}.\ \pi^k \in [\![\varphi_2]\!]_{\mathcal{K}}$ and $\forall j \in [0, k).\pi^j \in [\![\varphi_1]\!]_{\mathcal{K}}\}$;
- "Release": $[\![\mathrm{V}(\varphi_1, \varphi_2)]\!]_{\mathcal{K}} = \{\pi \in Path(\mathcal{K}) \mid \forall n \in \mathbb{N}.(\forall i \in [0, n).\pi^i \notin [\![\varphi_1]\!]_{\mathcal{K}}) \Rightarrow (\pi^n \in [\![\varphi_2]\!]_{\mathcal{K}})\}$.

"Globally" (G), "eventually" (F) and "weak-until" (W) can be defined as derived operators in LTL as follows: $\mathrm{G}\varphi \stackrel{\mathrm{def}}{=} \mathrm{V}(\mathit{false}, \varphi)$; $\mathrm{F}\varphi \stackrel{\mathrm{def}}{=} \mathrm{U}(\mathit{true}, \varphi)$; $\mathrm{W}(\varphi_1, \varphi_2) \stackrel{\mathrm{def}}{=} \mathrm{G}\varphi_1 \vee \mathrm{U}(\varphi_1, \varphi_2)$. Moreover, "release" can be expressed in terms of "weak-until": $\mathrm{V}(\varphi_1, \varphi_2) = \mathrm{W}(\varphi_2, \varphi_1 \wedge \varphi_2)$. If Op is any set of linear operators then we will denote by $\mathrm{L}(\mathit{Op})$ the subset of LTL formulae which are inductively generated by the grammar:

$$\mathrm{L}(\mathit{Op}) \ni \varphi ::= p \mid op(\varphi_1, ..., \varphi_n)$$

where op ranges over Op. The universal (or, dually, existential) path quantifier provides a state semantics of LTL. For any $\varphi \in \mathrm{LTL}$, $[\![\forall\varphi]\!]_{\mathcal{K}} = \{s \in \Sigma \mid \forall \pi \in Path(\mathcal{K}).\ (\pi(0) = s) \Rightarrow \pi \in [\![\varphi]\!]_{\mathcal{K}}\}$. For any $\mathrm{L} \subseteq \mathrm{LTL}$, $\forall\mathrm{L}$ denotes the set of universally quantified formulae of L, i.e. $\forall\mathrm{L} = \{\forall\varphi \mid \varphi \in \mathrm{L}\}$.

ACTL is defined by the following grammar:

$$\mathrm{ACTL} \ni \varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \mathrm{AX}\varphi \mid \mathrm{AU}(\varphi_1, \varphi_2) \mid \mathrm{AV}(\varphi_1, \varphi_2)$$

The standard semantics $[\![\cdot]\!]_{\mathcal{K}} : \mathrm{ACTL} \rightarrow \wp(\Sigma)$ w.r.t. a Kripke structure $\mathcal{K}$ goes as follows:

- $[\![p]\!]_{\mathcal{K}} = \{s \in \Sigma \mid p \in \ell(s)\}$;
- $[\![\neg p]\!]_{\mathcal{K}} = \Sigma \smallsetminus [\![p]\!]_{\mathcal{K}}$;
- $[\![\varphi_1 \wedge/\vee\ \varphi_2]\!]_{\mathcal{K}} = [\![\varphi_1]\!]_{\mathcal{K}} \cap/\cup\ [\![\varphi_2]\!]_{\mathcal{K}}$;
- $[\![\mathrm{AX}\varphi]\!]_{\mathcal{K}} = \{s \in \Sigma \mid \forall t \in \Sigma.\ (s \xrightarrow{R} t) \Rightarrow t \in [\![\varphi]\!]_{\mathcal{K}}\}$;
- $[\![\mathrm{AU}(\varphi_1, \varphi_2)]\!]_{\mathcal{K}} = \{s \in \Sigma \mid \forall \pi \in \mathrm{Path}(\mathcal{K}).\ (\pi(0) = s) \Rightarrow \exists k \in \mathbb{N}.\ \pi^k \in [\![\varphi_2]\!]_{\mathcal{K}}$ and $\forall j \in [0, k).\pi^j \in [\![\varphi_1]\!]_{\mathcal{K}}\}$;
- $[\![\mathrm{AV}(\varphi_1, \varphi_2)]\!]_{\mathcal{K}} = \{s \in \Sigma \mid \forall \pi \in \mathrm{Path}(\mathcal{K}).\ (\pi(0) = s) \Rightarrow (\forall n \in \mathbb{N}.\ (\forall i \in [0, n).\pi^i \notin [\![\varphi_1]\!]_{\mathcal{K}}) \Rightarrow (\pi^n \in [\![\varphi_2]\!]_{\mathcal{K}}))\}$.

It is well known that LTL, i.e. $\forall$LTL, and ACTL have incomparable expressive powers [2,8,13,20]. For instance, the ACTL formula AFAGp cannot be expressed in $\forall$LTL while the $\forall$LTL formula AFGp cannot be expressed in ACTL.

3 Abstract Interpretation of Inductive Languages

3.1 Abstract Interpretation Basics

In standard abstract interpretation, abstract domains can be equivalently specified either by *Galois connections/insertions* (GCs/GIs) or by (upper) *closure operators* (uco's) [4].

These two approaches are equivalent, modulo isomorphic representations of domain's objects. The closure operator approach has the advantage of being independent from the representation of domain's objects and is therefore appropriate for reasoning on abstract domains independently from their representation. Recall that $\mu : C \to C$ is a uco when μ is monotone, idempotent and extensive (viz. $x \leq \mu(x)$). It is well known that the set $\mathrm{uco}(C)$ of all uco's on C, endowed with the pointwise ordering $\sqsubseteq$, gives rise to the complete lattice $\langle \mathrm{uco}(C), \sqsubseteq \rangle$ of abstract domains of C. The ordering on $\mathrm{uco}(C)$ corresponds to the standard order which is used to compare abstract domains with regard to their precision: $\mu_1 \sqsubseteq \mu_2$ means that the domain μ_1 is a more precise abstraction of C than μ_2, or, equivalently, that the abstract domain μ_1 is a refinement of μ_2. Each closure $\mu \in \mathrm{uco}(C)$ is uniquely determined by the set $\mu(C)$ of its fixpoints, which is also its image $\mathrm{img}(\mu)$. Moreover, a subset $X \subseteq C$ is the set of fixpoints of a uco on C iff X is meet-closed (i.e. closed under arbitrary greatest lower bounds). Also, we have that $\mu \sqsubseteq \rho$ iff $\rho(C) \subseteq \mu(C)$. Often, we will identify closures with their sets of fixpoints since this does not give rise to ambiguity.

We denote by $\mathcal{G} = (\alpha, C, A, \gamma)$ a GC/GI of the abstract domain A into the concrete domain C through the abstraction and concretization maps α and γ forming an adjunction between C and A: $\alpha(c) \leq_C a \Leftrightarrow c \leq_A \gamma(a)$. Let us recall that it is enough to specify either the abstraction or the concretization map because in any GC the left adjoint map α determines the right adjoint map γ and vice versa: on the one hand, α is additive iff α admits the right adjoint $\gamma(a) = \vee_C\{c \in C \mid \alpha(c) \leq_A a\}$; on the other hand, γ is co-additive iff γ admits the left adjoint map $\alpha(c) = \wedge_A\{a \in A \mid c \leq_C \gamma(a)\}$. Recall that a GC is a GI when α is onto (or, equivalently, γ is 1-1), meaning that A does not contain useless abstract values. Recall that any GC $\mathcal{G}$ induces the uco $\mu_{\mathcal{G}} = \gamma \circ \alpha$ and conversely any $\mu \in \mathrm{uco}(C)$ induces a GI $(\mu, C, \mathrm{img}(\mu), \mathrm{id})$. Galois connections of a concrete domain C can be ordered according to their precision by exploiting the above ordering on the induced uco's: $\mathcal{G}_1 = (\alpha_1, C, A_1, \gamma_1) \leq \mathcal{G}_2 = (\alpha_2, C, A_2, \gamma_2)$ when $\mu_{\mathcal{G}_1} \sqsubseteq \mu_{\mathcal{G}_2}$. Let $\alpha : \wp(X) \to \wp(Y)$ and $\gamma : \wp(Y) \to \wp(X)$, and $\widetilde{\alpha} \overset{\text{def}}{=} \neg \circ \alpha \circ \neg$ and $\widetilde{\gamma} \overset{\text{def}}{=} \neg \circ \gamma \circ \neg$. Recall that $(\alpha, \wp(X)_{\subseteq/\supseteq}, \wp(Y)_{\subseteq/\supseteq}, \gamma)$ is a GC/GI iff $(\widetilde{\alpha}, \wp(X)_{\supseteq/\subseteq}, \wp(Y)_{\supseteq/\subseteq}, \widetilde{\gamma})$ is a GC/GI. Thus, results on α/γ and $\widetilde{\alpha}/\widetilde{\gamma}$ can be dualized through complementation.

By the above equivalence, throughout the paper, $\mathrm{uco}(C)_{\sqsubseteq}$ will play the role of the lattice of abstract interpretations of C [3,4], i.e. the complete lattice of all the abstract domains of the concrete domain C.

Let (α, C, A, γ) be a GI, $f : C \to C$ be some concrete semantic function — for simplicity of notation, we consider here 1-ary functions — and $f^\sharp : A \to A$ be a corresponding abstract semantic function. Then, $\langle A, f^\sharp \rangle$ is a sound abstract interpretation when $\alpha \circ f \sqsubseteq f^\sharp \circ \alpha$. The abstract function $f^A \overset{\text{def}}{=} \alpha \circ f \circ \gamma : A \to A$ is called the *best correct approximation* of f in A. *Completeness* in abstract interpretation [3,9] corresponds to require the following strengthening of soundness: $\alpha \circ f = f^\sharp \circ \alpha$. Hence, completeness corresponds to require that, in addition to soundness, no loss of precision is introduced by the abstract function $f^\sharp$ on the approximation $\alpha(c)$ of a concrete object $c \in C$ with respect to approximating by α the concrete computation $f(c)$. Completeness is an abstract domain property because it only depends on the abstract domain: in fact, it turns out that $\langle A, f^\sharp \rangle$ is complete iff $\langle A, f^A \rangle$ is complete. Thus, completeness can be equivalently stated as a property of closures as follows: $\mu \in \mathrm{uco}(C)$ is complete for f iff $\mu \circ f = \mu \circ f \circ \mu$ [9].

3.2 Abstract Semantics of Inductive Languages

Concrete Semantics. It is well known that abstract interpretation can be applied to approximate the semantics of any inductively defined language. Assume that formulae of a generic inductive language L are defined by:

$$\mathrm{L} \ni \varphi ::= p \mid f(\varphi_1, ..., \varphi_n)$$

where p ranges over a set of atomic propositions, that is left unspecified, while f ranges over a finite set Op of operators. Each operator $f \in Op$ has an arity[1] $\sharp(f) > 0$. The set of operators of L is also denoted by Op_{L}. Formulae in L are interpreted on a *semantic structure* $\mathcal{S} = (C, AP, I)$ where: C is any (concrete) domain of interpretation, AP is a set of atomic propositions and I is an interpretation function such that for any $p \in AP$, $I(p) \in C$ and for any $f \in Op$, $I(f) : C^{\sharp(f)} \to C$. For $p \in AP$ and $f \in Op$ we will also use $\boldsymbol{p}$ and $\boldsymbol{f}$ to denote, respectively, $I(p)$ and $I(f)$. Also, $\boldsymbol{Op} \stackrel{\mathrm{def}}{=} \{\boldsymbol{f} \mid f \in Op\}$. Hence, the *concrete semantic function* $[\![\cdot]\!]_{\mathcal{S}} : \mathrm{L} \to C$ is inductively defined as follows:

$$[\![p]\!]_{\mathcal{S}} = \boldsymbol{p} \quad \text{and} \quad [\![f(\varphi_1, ..., \varphi_n)]\!]_{\mathcal{S}} = \boldsymbol{f}([\![\varphi_1]\!]_{\mathcal{S}}, ..., [\![\varphi_n]\!]_{\mathcal{S}}).$$

When clear from the context, we will omit the subscript $\mathcal{S}$ which denotes the underlying semantic structure. The set of semantic evaluations in $\mathcal{S}$ of formulae in L is denoted by $\mathrm{Sem}_{\mathcal{S}}(\mathrm{L}) \stackrel{\mathrm{def}}{=} \{[\![\varphi]\!]_{\mathcal{S}} \mid \varphi \in \mathrm{L}\}$, or simply by $\mathrm{Sem}_C(\mathrm{L})$. $\mathrm{Sem}_C(\mathrm{L})$ is also called the *expressive power* (in C) of the language L.

If g is any syntactic operator with arity $\sharp(g) = n > 0$ and whose interpretation is given by $\boldsymbol{g} : C^n \to C$ then we say that a language L is *closed under* g when for any $\varphi_1, ..., \varphi_n \in \mathrm{L}$ there exists some $\psi \in \mathrm{L}$ such that $\boldsymbol{g}([\![\varphi_1]\!]_{\mathcal{S}}, ..., [\![\varphi_n]\!]_{\mathcal{S}}) = [\![\psi]\!]_{\mathcal{S}}$, for any semantic structure $\mathcal{S}$. In particular, if L is evaluated on a powerset $\wp(X)$ then L is closed under (infinite) logical conjunction iff for any $\Phi \subseteq \mathrm{L}$, there exists some $\psi \in \mathrm{L}$ such that $\bigcap_{\varphi \in \Phi} [\![\varphi]\!]_{\mathcal{S}} = [\![\psi]\!]_{\mathcal{S}}$.

The standard semantics of LTL and ACTL, as recalled in Section 2, can be viewed as concrete semantic functions, where the concrete semantic domains are given, respectively, by $\wp(\mathrm{Path}(\mathcal{K}))$ and $\wp(\Sigma)$.

Comparing Expressive Power. The standard notion of expressive power is used to compare different languages. Let L_1 and L_2 be two languages. L_1 is more expressive than L_2, denoted by $\mathrm{L}_1 \leq \mathrm{L}_2$, when for any semantic structure $\mathcal{S} = (C, AP, I)$ such that I provides an interpretation for all the operators in L_1 and L_2, $\mathrm{Sem}_{\mathcal{S}}(\mathrm{L}_2) \subseteq \mathrm{Sem}_{\mathcal{S}}(\mathrm{L}_1)$, while L_1 is equivalent to L_2, denoted by $\mathrm{L}_1 \equiv \mathrm{L}_2$, when $\mathrm{L}_1 \leq \mathrm{L}_2$ and $\mathrm{L}_2 \leq \mathrm{L}_1$, viz., $\mathrm{Sem}_{\mathcal{S}}(\mathrm{L}_1) = \mathrm{Sem}_{\mathcal{S}}(\mathrm{L}_2)$. For instance, as recalled in Section 2, ACTL and $\forall$LTL have incomparable expressive powers meaning that ACTL $\not\leq$ $\forall$LTL and $\forall$LTL $\not\leq$ ACTL.

Abstract Semantics. Within the standard abstract interpretation framework for defining abstract semantics [3,4], for a given semantic structure $\mathcal{S} = (C, AP, I)$, $C_{\leq}$ is a

[1] It would be possible to consider generic operators whose arity is any possibly infinite ordinal, thus allowing, for example, infinite conjunctions or disjunctions.

complete lattice which plays the role of concrete domain. Let us consider an abstract domain A specified by a GI $\mathcal{G} = (\alpha, C, A, \gamma)$. Thus, A induces an abstract semantic structure $\mathcal{S}^A = (A, AP, I^A)$ where I^A is defined through best correct approximations as follows:

$$I^A(p) \stackrel{\text{def}}{=} \alpha(I(p)) \quad \text{and} \quad I^A(f) \stackrel{\text{def}}{=} \alpha \circ I(f) \circ \gamma.$$

Thus, $\mathcal{S}^A$ induces the *abstract semantic function* $[\![\cdot]\!]^A_{\mathcal{S}} : \mathrm{L} \to A$ (also simply denoted by $[\![\cdot]\!]^A$). We will also use L^α or L^A to denote the abstract semantic evaluation of L induced by the abstract domain A so that $\mathrm{Sem}(\mathrm{L}^\alpha)$ (or $\mathrm{Sem}(\mathrm{L}^A)$) denotes the set of abstract semantics $\{[\![\varphi]\!]^A \mid \varphi \in \mathrm{L}\}$.

On the other hand, the domain A also induces the overall abstraction of the concrete semantics, namely ${}^A[\![\cdot]\!]_{\mathcal{S}} : \mathrm{L} \to A$ is defined by: ${}^A[\![\varphi]\!]_{\mathcal{S}} \stackrel{\text{def}}{=} \alpha([\![\varphi]\!]_{\mathcal{S}})$. In this case, we will use $\alpha\mathrm{L}$ to denote this abstract semantic evaluation of L induced by the abstract domain A so that $\mathrm{Sem}(\alpha\mathrm{L}) = \{{}^A[\![\varphi]\!]_{\mathcal{S}} \mid \varphi \in \mathrm{L}\}$.

Definition 1. The abstraction α is *complete* for L when $\mathrm{Sem}(\alpha\mathrm{L}) = \mathrm{Sem}(\mathrm{L}^\alpha)$.

This is indeed a generalization of Emerson and Halpern's [8] approach for comparing linear and branching formulae based on the universal path quantifier $\forall$. In fact, we will see in Section 4.2 how the universal path quantifier $\forall$ can be viewed as a particular abstraction of sets of traces, so that the branching language $\forall\mathrm{L} = \{\forall\varphi \mid \varphi \in \mathrm{L}\}$ and the corresponding results in [8] can be cast as particular cases in our framework.

It turns out that the abstract semantic function is always sound by construction: for any $\varphi \in \mathrm{L}$, $\alpha([\![\varphi]\!]_{\mathcal{S}}) \leq_A [\![\varphi]\!]^A_{\mathcal{S}}$ (or, equivalently, $[\![\varphi]\!]_{\mathcal{S}} \leq_C \gamma([\![\varphi]\!]^A_{\mathcal{S}})$). As far as completeness is concerned, it turns out that completeness of the abstract domain A for (the interpretation of) the operators in *Op* ensures completeness of the abstract semantic function [5].

Theorem 1 (Cousot and Cousot [5]). *If A is complete for every $\boldsymbol{f} \in \boldsymbol{Op}_{\mathrm{L}}$ then for every $\varphi \in \mathrm{L}$, $\alpha([\![\varphi]\!]_{\mathcal{S}}) = [\![\varphi]\!]^A_{\mathcal{S}}$. In this case, α is complete for* L.

4 Abstracting Traces

4.1 Trace Semantics of Linear Languages

As recalled above, the standard semantics of a linear formula $\varphi \in \mathrm{LTL}$ consists of a set of paths in a Kripke structure $\mathcal{K} = (\Sigma, R, AP, \ell)$. $\mathrm{Path}(\mathcal{K})$ can be viewed as a *model* M for interpreting LTL. It turns out that this standard semantics can be obtained as an abstract interpretation of a more general semantics which evaluates formulae in LTL as a set of traces and therefore is independent from a given model. Following [5], this *trace semantics* $[\![\cdot]\!] : \mathrm{LTL} \to \wp(\mathrm{Trace}(\Sigma))$ only depends on a state space Σ and is as follows:

- $[\![p]\!] = \{\sigma \in \mathrm{Trace}(\Sigma) \mid p \in \ell(\sigma(0))\}$;
- $[\![\neg p]\!] = \mathrm{Trace}(\Sigma) \smallsetminus [\![p]\!]$;
- $[\![\varphi_1 \wedge/\vee\ \varphi_2]\!] = [\![\varphi_1]\!] \cap/\cup\ [\![\varphi_2]\!]$;
- $[\![\mathrm{X}\varphi]\!] = \mathbf{X}([\![\varphi]\!]) \stackrel{\text{def}}{=} \{\sigma \in \mathrm{Trace}(\Sigma) \mid \sigma^1 \in [\![\varphi]\!]\}$;

– $[\![\mathrm{U}(\varphi_1, \varphi_2)]\!] = \mathbf{U}([\![\varphi_1, \varphi_2]\!]) \stackrel{\text{def}}{=} \{\sigma \in \mathrm{Trace}(\Sigma) \mid \exists k \in \mathbb{N}.\, \sigma^k \in [\![\varphi_2]\!]$ and $\forall j \in [0, k).\sigma^j \in [\![\varphi_1]\!]\}$;
– $[\![\mathrm{V}(\varphi_1, \varphi_2)]\!] = \mathbf{V}([\![\varphi_1, \varphi_2]\!]) \stackrel{\text{def}}{=} \{\sigma \in \mathrm{Trace}(\Sigma) \mid \forall n \in \mathbb{N}.\, (\forall i \in [0, n).\sigma^i \notin [\![\varphi_1]\!]) \Rightarrow (\sigma^n \in [\![\varphi_2]\!])\}$.

Let $M = Path(\mathcal{K})$ be a model and let us define $\alpha_{M_\forall} : \wp(\mathrm{Trace}(\Sigma)) \to \wp(M)$ and $\gamma_{M_\forall} : \wp(M) \to \wp(\mathrm{Trace}(\Sigma))$ as follows:

$$\alpha_{M_\forall}(T) \stackrel{\text{def}}{=} T \cap M \quad \text{and} \quad \gamma_{M_\forall}(P) \stackrel{\text{def}}{=} P.$$

It is easy to note that $(\alpha_{M_\forall}, \wp(\mathrm{Trace}(\Sigma))_\supseteq, \wp(M)_\supseteq, \gamma_{M_\forall})$ is a GI. This is a *universal model abstraction* (hence the subscript $\forall$) because sets of traces and paths are ordered by superset inclusion. The abstraction map $\lambda T.T \cap M$ on $\wp(\mathrm{Trace}(\Sigma))_\subseteq$ gives also rise to the *existential model abstraction* $(\alpha_{M_\exists}, \wp(\mathrm{Trace}(\Sigma))_\subseteq, \wp(M)_\subseteq, \gamma_{M_\exists})$ where:

$$\alpha_{M_\exists}(T) \stackrel{\text{def}}{=} T \cap M \quad \text{and} \quad \gamma_{M_\exists}(P) \stackrel{\text{def}}{=} P \cup \neg M.$$

Note that $\neg M$ is the set of "spurious" traces, namely traces that are not paths. This is a GI as well. Existential abstraction is dual to universal abstraction because: $\gamma_{M_\exists} \circ \alpha_{M_\exists} = \neg \circ (\gamma_{M_\forall} \circ \alpha_{M_\forall}) \circ \neg$.

It is immediate to notice that for any $\varphi \in \mathrm{LTL}$, $[\![\varphi]\!]_{\mathcal{K}} = \alpha_{M_\forall}([\![\varphi]\!]) = \alpha_{M_\exists}([\![\varphi]\!])$. In abstract interpretation terms, this is a consequence of the fact that the standard path semantics of LTL is a complete abstract interpretation of trace semantics.

Proposition 1. *$\alpha_{M_\forall}$ and $\alpha_{M_\exists}$ are complete for the linear operators in $\boldsymbol{Op}_{\mathrm{LTL}}$.*

As a consequence, $\alpha_{M_\forall}([\![\varphi]\!]) = [\![\varphi]\!]^{\alpha_{M_\forall}} = [\![\varphi]\!]_{\mathcal{K}}$ and $\alpha_{M_\exists}([\![\varphi]\!]) = [\![\varphi]\!]^{\alpha_{M_\exists}} = [\![\varphi]\!]_{\mathcal{K}}$, namely path semantics can be retrieved as complete abstractions of trace semantics.

In the following, we provide a number of abstractions of the trace semantics of LTL, based on abstract domains of the existential/universal concrete domain $\wp(\mathrm{Trace}(\Sigma))_{\subseteq/\supseteq}$. Any such trace abstraction $\alpha_M^{\exists/\forall} : \wp(\mathrm{Trace}(\Sigma))_{\subseteq/\supseteq} \to A$ depends on a model $M = Path(\mathcal{K})$ and can be factorized as $\alpha_M^{\exists/\forall} = \alpha^{\exists/\forall} \circ \alpha_{M_{\exists/\forall}}$, where $\alpha^{\exists/\forall} : \wp(M)_{\subseteq/\supseteq} \to A$ is a path abstraction, namely an abstraction of of the existential/universal domain $\wp(M)_{\subseteq/\supseteq}$ of sets of paths. It turns out that the above Proposition 1 makes completeness of trace and path abstractions $\alpha_M^{\exists/\forall}$ and $\alpha^{\exists/\forall}$ equivalent. In fact, if $\boldsymbol{f} : \wp(\mathrm{Trace}(\Sigma)) \to \wp(\mathrm{Trace}(\Sigma))$ is a linear trace operator and $\boldsymbol{f}^{\exists/\forall} : \wp(M) \to \wp(M)$ is the corresponding linear path operator induced by $\alpha_{M_{\exists/\forall}}$ (i.e., $\boldsymbol{f}^{\exists/\forall} = \alpha_{M_{\exists/\forall}} \circ \boldsymbol{f} \circ \gamma_{M_{\exists/\forall}}$), then $\alpha_M^{\exists/\forall}$ is complete for $\boldsymbol{f}$ iff $\alpha^{\exists/\forall}$ is complete for $\boldsymbol{f}^{\exists/\forall}$.

4.2 Branching Abstraction

As shown by Cousot and Cousot [5], the universal path quantifier allows to cast states as an abstraction of traces, so that state-based model checking can be viewed as an abstraction of trace-based model checking. Let $\mathcal{K}$ be a Kripke structure and $M = Path(\mathcal{K})$ be the corresponding model. For any $s \in \Sigma$ and $i \in \mathbb{N}$, we define $M^i_{\downarrow s} \stackrel{\text{def}}{=} \{\pi \in M \mid \pi(i) = s\}$. Therefore, $M^i_{\downarrow s}$ is the set of paths in M whose state at time i is s. In particular, $M^0_{\downarrow s}$ is the set of paths that start in s. The *universal branching abstraction* $\mathcal{G}^\forall = (\alpha^\forall_M, \wp(\mathrm{Trace}(\Sigma))_\supseteq, \wp(\Sigma)_\supseteq, \gamma^\forall_M)$ is defined as follows:

$$\alpha^\forall_M(T) \stackrel{\text{def}}{=} \{s \in \Sigma \mid M^0_{\downarrow s} \subseteq T\} \quad \text{and} \quad \gamma^\forall_M(S) \stackrel{\text{def}}{=} \{\pi \in M \mid \pi(0) \in S\}.$$

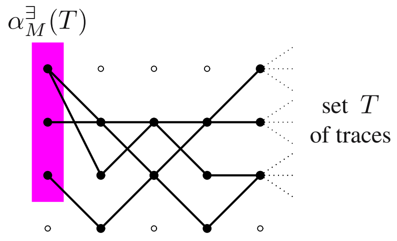

Fig. 1. Branching abstraction

It turns out that $\mathcal{G}^{\forall}$ is a GI. The *existential* branching abstraction is defined by duality:

– $\alpha^{\exists}_{M}(T) \stackrel{\text{def}}{=} \neg(\alpha^{\forall}_{M}(\neg(T))) = \{s \in \Sigma \mid M^{0}_{\downarrow s} \cap T \neq \varnothing\}$;
– $\gamma^{\exists}_{M}(S) \stackrel{\text{def}}{=} \neg(\gamma^{\forall}_{M}(\neg(S))) = \{\pi \in \wp(\text{Trace}(\Sigma)) \mid (\pi \in M) \Rightarrow (\pi(0) \in S)\}$.

In this case, $\mathcal{G}^{\exists} = (\alpha^{\exists}_{M}, \wp(\text{Trace}(\Sigma))_{\subseteq}, \wp(\Sigma)_{\subseteq}, \gamma^{\exists}_{M})$ is a GI. An example of existential branching abstraction is depicted in Figure 1.

The branching abstraction exactly formalizes the universal path quantification of LTL formulae: in fact, it is immediate to observe that for any $\varphi \in$ LTL, $[\![\forall\varphi]\!]_{\mathcal{K}} = \alpha^{\forall}_{M}([\![\varphi]\!]_{\mathcal{K}})$. Moreover, as shown by Cousot and Cousot [5], it turns out that the branching abstraction $\text{LTL}^{\alpha^{\forall}_{M}}$ of LTL exactly gives ACTL, namely the best correct approximations of the linear operators of LTL induced by $\alpha^{\forall}_{M}$ coincide with the branching state temporal operators of ACTL. Therefore, $\text{Sem}(\text{LTL}^{\alpha^{\forall}_{M}}) = \text{Sem}(\text{ACTL})$.

As recalled above, it is well known that ∀LTL and ACTL have incomparable expressive powers. In our framework, this means that $\text{Sem}(\alpha^{\forall}_{M}\text{LTL})$ and $\text{Sem}(\text{LTL}^{\alpha^{\forall}_{M}}) = \text{Sem}(\text{ACTL})$ are incomparable sets, i.e. the branching abstraction is incomplete for LTL. As a consequence, by Theorem 1, it turns out that the branching abstraction is incomplete for some operators in $\boldsymbol{Op}_{\text{LTL}}$. The sources of incompleteness of $\alpha^{\forall}_{M}$ have been analyzed by Cousot and Cousot [5]: the branching abstraction results to be incomplete for the disjunction, until and release operators (see [5]). On the other hand, Maidl [14] provides a synctatic characterization of the maximum common fragment, called LTL_{det}, of LTL and ACTL: for any $\varphi \in$ LTL, $\alpha^{\forall}_{M}([\![\varphi]\!]) \in \text{Sem}(\text{ACTL})$ iff $[\![\varphi]\!] \in \text{Sem}(\text{LTL}_{\text{det}})$. We will further discuss completeness of the branching abstraction in Section 5.1.

4.3 Trace of Sets Abstraction

Sets of traces can be approximated by a trace of sets. Let us formalize this approximation. We consider the abstract domain $\text{Trace}(\wp(\Sigma))$, namely sequences of sets of states. Traces of sets are ordered pointwise: if $\sigma, \tau \in \text{Trace}(\wp(\Sigma))$ then $\sigma \sqsubseteq \tau$ iff $\forall i \in \mathbb{N}.\ \sigma(i) \subseteq \tau(i)$. Thus, we first consider existential traces of sets because $\wp(\Sigma)$ is here ordered by $\subseteq$. It turns out that $\text{Trace}(\wp(\Sigma))_{\sqsubseteq}$ is a complete lattice where $\sigma \sqcup \tau = \lambda i.\ \sigma(i) \cup \tau(i)$ and $\sigma \sqcap \tau = \lambda i.\ \sigma(i) \cap \tau(i)$. The *existential trace of sets abstraction* $\alpha^{\exists t}_{M} : \wp(\text{Trace}(\Sigma)) \to \text{Trace}(\wp(\Sigma))$ is then defined as follows:

$$\alpha^{\exists t}_{M}(T) \stackrel{\text{def}}{=} \lambda i \in \mathbb{N}.\{\pi(i) \mid \pi \in T \cap M\} = \lambda i \in \mathbb{N}.\{s \in \Sigma \mid M^{i}_{\downarrow s} \cap T \neq \varnothing\}$$

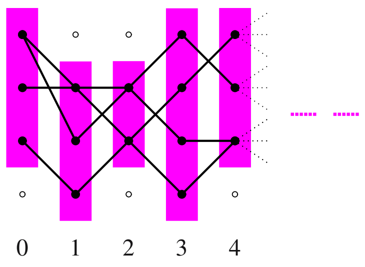

Fig. 2. Existential trace of sets abstraction

$$\begin{tikzcd} a \arrow[r] & b \arrow[loop right] \end{tikzcd} \qquad \begin{tikzcd} a \arrow[r] \arrow[dr] & b \arrow[loop right] \\ c \arrow[r] \arrow[ur] & d \arrow[loop right] \end{tikzcd} \qquad \begin{tikzcd} a \arrow[r, bend left] & b \arrow[l, bend left] \arrow[loop above] & c \arrow[l] \end{tikzcd}$$

Fig. 3. Transition systems $\mathcal{T}_1$ (left), $\mathcal{T}_2$ (middle) and $\mathcal{T}_3$ (right)

and together with its adjoint map $\gamma_M^{\exists t} : \mathrm{Trace}(\wp(\Sigma)) \to \wp(\mathrm{Trace}(\Sigma))$ defined by

$$\gamma_M^{\exists t}(\tau) \stackrel{\text{def}}{=} \{\pi \in \mathrm{Trace}(\Sigma) \mid (\pi \in M) \Rightarrow \forall i \in \mathbb{N}.\, \pi(i) \in \tau(i)\}$$

gives rise to a GC.

Theorem 2. *$\mathcal{G}^{\exists t} = (\alpha_M^{\exists t}, \wp(\mathrm{Trace}(\Sigma))_{\subseteq}, \mathrm{Trace}(\wp(\Sigma))_{\sqsubseteq}, \gamma_M^{\exists t})$ is a GC.*

A graphical example of an existential trace of sets abstraction is given in Figure 2. It turns out that $\mathcal{G}^{\exists t}$ is not a GI. In fact, for the transition system $\mathcal{T}_1$ in Figure 3, $\gamma_M^{\exists t}$ is not 1-1: $\gamma_M^{\exists t}(\langle\{a\},\{a\},\{b\},\{b\},\{b\},\ldots\rangle) = \gamma_M^{\exists t}(\langle\{a\},\{a\},\{a\},\{b\},\{b\},\ldots\rangle) = \neg M$.

The *universal* trace of sets abstraction is dually defined, where the complement of a trace of sets is defined pointwise, namely for any $\tau \in \mathrm{Trace}(\wp(\Sigma))$, $\neg\tau = \lambda i.\, \neg\tau(i)$.

- $\alpha_M^{\forall t}(T) \stackrel{\text{def}}{=} \neg(\alpha_M^{\exists t}(\neg(T))) = \lambda i \in \mathbb{N}.\{s \in \Sigma \mid M_{\downarrow s}^{i} \subseteq T\}$;
- $\gamma_M^{\forall t}(\tau) \stackrel{\text{def}}{=} \neg(\gamma_M^{\exists t}(\neg\tau)) = \{\pi \in M \mid \exists i \in \mathbb{N}.\pi(i) \in \tau(i)\}$.

Hence, $\mathcal{G}^{\forall t} = (\alpha_M^{\forall t}, \wp(\mathrm{Trace}(\Sigma))_{\supseteq}, \mathrm{Trace}(\wp(\Sigma))_{\sqsupseteq}, \gamma_M^{\forall t})$ is a GC as well.

4.4 Reachable State Abstraction

One can approximate a set T of traces through the set of states that can be reached by some path in T. Let us formalize this approximation. For any $s \in \Sigma$, let us define $M_{\downarrow s} \stackrel{\text{def}}{=} \bigcup_{i \in \mathbb{N}} M_{\downarrow s}^{i} = \{\pi \in M \mid \exists i \in \mathbb{N}.\, \pi(i) = s\}$. Thus, the *existential reachable state abstraction* $\alpha_M^{\exists r} : \wp(\mathrm{Trace}(\Sigma)) \to \wp(\Sigma)$ is defined by:

$$\alpha_M^{\exists r}(T) \stackrel{\text{def}}{=} \{s \in \Sigma \mid M_{\downarrow s} \cap T \neq \varnothing\}.$$

Therefore, the corresponding concretization map $\gamma_M^{\exists r} : \wp(\Sigma) \to \wp(\mathrm{Trace}(\Sigma))$ is as follows:

$$\gamma_M^{\exists r}(S) \stackrel{\text{def}}{=} \{\pi \in \mathrm{Trace}(\Sigma) \mid (\pi \in M) \Rightarrow (\forall i \in \mathbb{N}.\, \pi(i) \in S)\}.$$

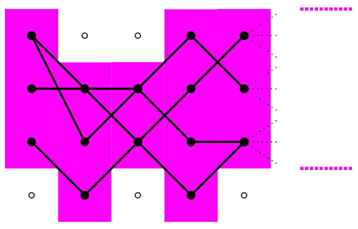

Fig. 4. Existential reachable state abstraction

Theorem 3. $\mathcal{G}^{\exists r} \stackrel{\text{def}}{=} (\alpha_M^{\exists r}, \wp(\text{Trace}(\Sigma))_{\subseteq}, \wp(\Sigma)_{\subseteq}, \gamma_M^{\exists r})$ *is a GC.*

A graphical example of existential reachable state abstraction is depicted in Figure 4. Also in this case, this is not a GI. In fact, by considering the transition system $\mathcal{T}_1$ in Figure 3, we have that $\gamma_M^{\exists r}$ is not 1-1: $\gamma_M^{\exists r}(\varnothing) = \gamma_M^{\exists r}(\{a\}) = \neg M$.

By duality, the *universal* reachable state abstraction is defined as follows:

– $\alpha_M^{\forall r}(T) \stackrel{\text{def}}{=} \neg(\alpha_M^{\exists r}(\neg(T))) = \lambda i \in \mathbb{N}.\{s \in \Sigma \mid M_{\downarrow s} \subseteq T\}$;
– $\gamma_M^{\forall r}(S) \stackrel{\text{def}}{=} \neg(\gamma_M^{\exists r}(\neg S)) = \{\pi \in M \mid \exists i \in \mathbb{N}.\pi(i) \in S\}$.

Hence, $\mathcal{G}^{\forall r} = (\alpha_M^{\forall r}, \wp(\text{Trace}(\Sigma))_{\supseteq}, \text{Trace}(\wp(\Sigma))_{\supseteq}, \gamma_M^{\forall r})$ is a GC as well.

4.5 Comparing Trace Abstractions

It turns out that traces of sets of states are more precise than both branching and reachable states, while branching and reachable states abstractions are incomparable.

Proposition 2. $\mathcal{G}^{\forall t} \leq \mathcal{G}^{\forall}$ *and* $\mathcal{G}^{\forall t} \leq \mathcal{G}^{\forall r}$. *Also,* $\mathcal{G}^{\forall}$ *and* $\mathcal{G}^{\forall r}$ *are incomparable.*

5 Completeness of Trace Abstractions

5.1 Branching Abstraction

The maximum common fragment LTL_{det} of LTL and ACTL has been characterized by Maidl [14]:

$$\text{LTL}_{\text{det}} \ni \varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid (p \wedge \varphi_1) \vee (\neg p \wedge \varphi_2) \mid$$
$$\text{X}\varphi \mid \text{U}(p \wedge \varphi_1, \neg p \wedge \varphi_2) \mid \text{W}(p \wedge \varphi_1, \neg p \wedge \varphi_2)$$

Maidl [14] shows that $\text{LTL}_{\text{det}} = \text{LTL} \cap \text{ACTL}$, namely for any $\varphi \in \text{LTL}$, $\alpha_M^{\forall}([\![\varphi]\!]) \in \text{Sem}(\text{ACTL})$ iff $[\![\varphi]\!] \in \text{Sem}(\text{LTL}_{\text{det}})$. This result is important in abstract model checking because formulae in $\text{LTL} \cap \text{ACTL}$ admit linear counterexamples.

It turns out that the branching abstraction is complete for all the logical/temporal operators of LTL_{det}.

Theorem 4. $\alpha_M^{\forall}$ *is complete for the logical/linear operators in* $\boldsymbol{Op}_{\mathrm{LTL_{det}}}$.

Thus, as expected, by Theorem 1 we obtain that the branching abstraction is complete for $\mathrm{LTL_{det}}$ so that trace-based and state-based model checking of $\mathrm{LTL_{det}}$ are equivalent.

We also obtain a further consequence from this completeness result. Some attempts prior to Maidl's 2000 work [14] of characterizing LTL∩ACTL considered the so-called *branchable* formulae of LTL: given $\varphi \in \mathrm{LTL}$, the formula $\varphi_{\mathrm{A}} \in \mathrm{ACTL}$ is obtained from φ by preceding each linear temporal operator occurring in φ by the universal path quantifier A. A formula $\varphi \in \mathrm{LTL}$ is branchable when $\alpha_M^{\forall}(\llbracket\varphi\rrbracket) = \llbracket\varphi_{\mathrm{A}}\rrbracket$ [11,20]. We thus define $\mathrm{LTL_{br}} \stackrel{\mathrm{def}}{=} \{\varphi \in \mathrm{LTL} \mid \varphi \text{ is branchable}\}$. Since the abstract semantics $\llbracket\varphi\rrbracket^{\alpha_M^{\forall}}$ of a LTL formula φ exactly coincides with $\llbracket\varphi_{\mathrm{A}}\rrbracket$, by Theorem 4, we have that $\mathrm{LTL_{br}} = \{\varphi \in \mathrm{LTL} \mid \alpha_M^{\forall}(\llbracket\varphi\rrbracket) = \llbracket\varphi\rrbracket^{\alpha_M^{\forall}}\}$. As a consequence of the above Theorem 4, of Theorem 1 and of Maidl's Theorem, we obtain the following alternative characterization relating $\mathrm{LTL_{det}}$ and $\mathrm{LTL_{br}}$.

Theorem 5. $\mathrm{Sem}(\mathrm{LTL_{det}}) = \mathrm{Sem}(\mathrm{LTL_{br}})$.

5.2 Trace of Sets Abstraction

The trace of sets abstraction $\alpha_M^{\forall t}$ is not complete for all the operators of LTL. This is indeed a consequence of a more general result in [16] stating that no refinement of the branching abstraction can be complete for all the operators of LTL. As an example, let us show that $\alpha_M^{\forall t}$ is not complete for disjunction. In fact, for the transition system $\mathcal{T}_2$ in Figure 3, by considering the set of traces (actually paths) $T_1 = \{ab^\omega, cd^\omega\}$ and $T_2 = \{ad^\omega, cb^\omega\}$, we have that:

$$\begin{aligned}
\alpha_M^{\forall t}(\neg T_1) \sqcup \alpha_M^{\forall t}(\neg T_2) &= \\
\langle\{b,d\},\{a,c\},\{a,c\},...\rangle \sqcup \langle\{b,d\},\{a,c\},\{a,c\},...\rangle &= \\
\langle\{b,d\},\{a,c\},\{a,c\},...\rangle &\sqsubset \\
\alpha_M^{\forall t}(\neg T_1 \cup \neg T_2) &= \\
\alpha_M^{\forall t}(\mathrm{Trace}(\Sigma)) &= \\
\langle\{a,b,c,d\},\{a,b,c,d\},\{a,b,c,d\},...\rangle &
\end{aligned}$$

On the other hand, it turns out that traces of sets are complete for conjunction, next and globally operators.

Proposition 3. $\alpha_M^{\forall t}$ *is complete for the following operators on* $\wp(\mathrm{Trace}(\Sigma))$: $\lambda S,T.S \cap T$, $\lambda T.\mathbf{X}(T)$, $\lambda T.\mathbf{G}(T)$.

Thus, by Theorem 1, we obtain that $\alpha_M^{\forall t}$ is complete for $\mathrm{L}(\{\wedge, \mathrm{X}, \mathrm{G}\})$. By duality, $\alpha_M^{\exists t}$ is complete for $\mathrm{L}(\{\vee, \mathrm{X}, \mathrm{F}\})$.

5.3 Reachable State Abstraction

As expected, also the reachable states abstraction is not complete for all the linear operators of LTL. For instance, let us show that $\alpha_M^{\forall r}$ is not complete for disjunction and next operators. As far as disjunction is concerned, let us consider the transition system $\mathcal{T}_3$ in Figure 3 and the set of traces (actually paths) $T_1 = \{ab^\omega\}$ and $T_2 = \{cb^\omega\}$.

$$\alpha_M^{\forall r}(\neg T_1) \cup \alpha_M^{\forall r}(\neg T_2) = \{c\} \cup \{a\} \neq \alpha_M^{\forall r}(\neg T_1 \cup \neg T_2) = \alpha_M^{\forall r}(\mathrm{Trace}(\Sigma)) = \{a,b,c\}.$$

Moreover, for the next operator $\mathbf{X}$ we have that:

$$\alpha_M^{\forall r}(\mathbf{X}(\neg\{ab^\omega\})) = \neg\alpha_M^{\exists r}(\mathbf{X}(\{ab^\omega\})) = \neg\alpha_M^{\exists r}(\{aab^\omega, bab^\omega, cab^\omega\}) = \neg\{a,b\} = \{c\}.$$

Conversely, we have that $\alpha_M^{\forall r}(\mathbf{X}(\gamma_M^{\forall r}(\alpha_M^{\forall r}(\neg\{ab^\omega\})))) = \neg\alpha_M^{\exists r}(\mathbf{X}(\gamma_M^{\exists r}(\alpha_M^{\exists r}(\{ab^\omega\}))))$ and $b^\omega \in \gamma_M^{\exists r}(\alpha_M^{\exists r}(\{ab^\omega\}))$ so that $cb^\omega \in \mathbf{X}(\gamma_M^{\exists r}(\alpha_M^{\exists r}(\{ab^\omega\})))$ and in turn $c \in \alpha_M^{\exists r}(\mathbf{X}(\gamma_M^{\exists r}(\alpha_M^{\exists r}(\{ab^\omega\}))))$, namely $c \notin \alpha_M^{\forall r}(\mathbf{X}(\gamma_M^{\forall r}(\alpha_M^{\forall r}(\neg\{ab^\omega\}))))$. We have thus shown incompleteness for $\mathbf{X}$:

$$\alpha_M^{\forall r}(\mathbf{X}(\neg\{ab^\omega\})) \neq \alpha_M^{\forall r}(\mathbf{X}(\gamma_M^{\forall r}(\alpha_M^{\forall r}(\neg\{ab^\omega\})))).$$

Here, it turns out that $\alpha_M^{\forall r}$ is complete for conjunction and globally operators.

Proposition 4. *$\alpha_M^{\forall r}$ is complete for the following operators on $\wp(\mathrm{Trace}(\Sigma))$: $\lambda S, T. S \cap T$ and $\lambda T.\mathbf{G}(T)$.*

Therefore, by Theorem 1, we obtain that $\alpha_M^{\forall r}$ is complete for $\mathrm{L}(\{\wedge, \mathrm{G}\})$. By duality, $\alpha_M^{\exists r}$ is complete for $\mathrm{L}(\{\vee, \mathrm{F}\})$.

6 Comparing Expressive Powers

The standard notion of expressive power recalled in Section 3.2 is based on the idea of comparing languages in a common domain of interpretation. In fact, given a domain of interpretation C of some semantic structure $\mathcal{S} = (C, AP, I)$ we can compare two languages L_1 and L_2 by comparing $\mathrm{Sem}_C(\mathrm{L}_1)$ and $\mathrm{Sem}_C(\mathrm{L}_2)$. Thus, we write $\mathrm{L}_1 \equiv_C \mathrm{L}_2$ ($\mathrm{L}_1 \leq_C / \geq_C \mathrm{L}_2$) when $\mathrm{Sem}_C(\mathrm{L}_1) = \mathrm{Sem}_C(\mathrm{L}_2)$ ($\mathrm{Sem}_C(\mathrm{L}_1) \supseteq / \subseteq \mathrm{Sem}_C(\mathrm{L}_2)$).

More in general, we can also compare languages in a common "abstract" domain of interpretation. For example, it is well known how to compare state languages in the abstract domain of partitions. Let L be a state language, namely to be evaluated on $\wp(\Sigma)$. Following Dams [6,7], the *logical equivalence* $\sim_{\mathrm{L}}$ on the state space Σ induced by L is defined as follows: $s_1 \sim_{\mathrm{L}} s_2$ iff $\forall \varphi \in \mathrm{L}. s_1 \in [\![\varphi]\!] \Leftrightarrow s_2 \in [\![\varphi]\!]$. The state partition associated to the equivalence $\sim_{\mathrm{L}}$ is here denoted by $P_{\mathrm{L}} \in \mathrm{Part}(\Sigma)$ and, following Dams [6,7], is called the *distinguishing power* of L. Then, two state languages L_1 and L_2 can be also compared according to their distinguishing power:

$$\mathrm{L}_1 \equiv_{\mathrm{Part}(\Sigma)} \mathrm{L}_2 \quad \text{iff} \quad P_{\mathrm{L}_1} = P_{\mathrm{L}_2}.$$

Of course, this is indeed an "abstract" way of comparing languages because

$$\mathrm{L}_1 \equiv_{\wp(\Sigma)} \mathrm{L}_2 \;\Rightarrow\; \mathrm{L}_1 \equiv_{\mathrm{Part}(\Sigma)} \mathrm{L}_2$$

while the reverse implication is obviously not true. The distinguishing power is more abstract than the expressive power because it is not able to discriminate presence/absence of negation: in fact, for any language L, if $\mathrm{L}^\neg$ denotes the language L plus negation then we have that $P_{\mathrm{L}} = P_{\mathrm{L}^\neg}$ [17]. As an example, let L_μ denotes mu-calculus. It is well known [1] that $P_{\mathrm{CTL}} = P_{\mathrm{L}_\mu}$, that is $\mathrm{CTL} \equiv_{\mathrm{Part}(\Sigma)} \mathrm{L}_\mu$, whereas $\mathrm{Sem}_{\wp(\Sigma)}(\mathrm{CTL}) \subsetneq \mathrm{Sem}_{\wp(\Sigma)}(\mathrm{L}_\mu)$, that is $\mathrm{L}_\mu \lneq_{\wp(\Sigma)} \mathrm{CTL}$. A number of further examples can be found in Dams' works [6,7].

The key point here is that the lattice $\mathrm{Part}(\Sigma)$ of state partitions is indeed an abstraction of the lattice of abstract domains $\mathrm{uco}(\wp(\Sigma))$, as shown in [17]. Starting from this observation, the idea of comparing languages at different levels of abstraction can be precisely formalized by abstract interpretation. Let us recall from [17] how $\mathrm{Part}(\Sigma)$ can be viewed as an abstraction of the lattice of abstract domains $\mathrm{uco}(\wp(\Sigma))$. We define the abstraction and concretization maps:

$$\mathrm{uco}(\wp(\Sigma)_{\subseteq})_{\sqsupseteq} \underset{\mathrm{par}}{\overset{\mathrm{pcl}}{\leftrightarrows}} \mathrm{Part}(\Sigma)_{\succeq}$$

where, for any $s \in \Sigma$ and $\mu \in \mathrm{uco}(\wp(\Sigma))$, $[s]_\mu \stackrel{\mathrm{def}}{=} \{s' \in \Sigma \mid \mu(\{s'\}) = \mu(\{s\})\}$ and $\mathrm{par}(\mu) \stackrel{\mathrm{def}}{=} \{[s]_\mu \mid s \in \Sigma\}$, while $\mathrm{pcl}(P) \stackrel{\mathrm{def}}{=} \lambda X \in \wp(\Sigma). \cup \{B \in P \mid X \cap B \neq \varnothing\}$. Thus, two states belong to the same block of $\mathrm{par}(\mu)$ when they are abstracted by μ to the same set while $\mathrm{pcl}(P)(X)$ is the minimal covering of the set $X \subseteq \Sigma$ through blocks in P.Let us also remark that $\mathrm{pcl}(P)$ is a uco whose set of fixpoints is given by all the unions of blocks in P, i.e. $\mathrm{pcl}(P) = \{\cup_i B_i \mid \{B_i\} \subseteq P\}$. It turns out that $(\mathrm{par}, \mathrm{uco}(\wp(\Sigma))_{\sqsupseteq}, \mathrm{Part}(\Sigma)_{\succeq}, \mathrm{pcl})$ is a GI.

Let us observe that a state language L (to be evaluated on $\wp(\Sigma)$) which is closed under conjunction can be viewed as an abstract domain, in the sense that $\mathrm{Sem}_{\wp(\Sigma)}(\mathrm{L}) \in \mathrm{uco}(\wp(\Sigma)_{\subseteq})$ because $\mathrm{Sem}_{\wp(\Sigma)}(\mathrm{L})$ is meet-closed (cf. Section 3.1). Assume now that L is closed under conjunction. Then, the distinguishing power of L can be retrieved as an abstraction in $\mathrm{Part}(\Sigma)$ of the expressive power of L, that is $P_{\mathrm{L}} = \mathrm{par}(\mathrm{Sem}_{\wp(\Sigma)})$.

Obviously, this can be done in general for *any abstraction* $(\alpha, \mathrm{uco}(\wp(\Sigma))_{\sqsupseteq}, A, \gamma)$, namely we can define the α-expressive power of a state language L as the abstract value $\alpha(\mathrm{Sem}_{\wp(\Sigma)}) \in A$. Notice that $(\alpha, \mathrm{uco}(\wp(\Sigma))_{\sqsupseteq}, A, \gamma)$ is a generic *higher-order* abstract interpretation meaning that here we deal with an abstraction of the higher-order lattice of abstract domains of the concrete domain $\wp(\Sigma)$.

6.1 Generalizing the Linear vs. Branching Time Comparison

As recalled in Section 3.2, Emerson and Halpern [8] use the universal path quantifier $\forall$ for "abstracting" a linear time language L to a corresponding branching time language $\mathrm{B(L)} \stackrel{\mathrm{def}}{=} \{\forall\varphi \mid \varphi \in \mathrm{L}\}$ so that the expressive power of L can be compared with that of any state language. As shown in Section 4.2, the path quantifier $\forall$ can be cast as the branching abstract interpretation $(\alpha_M^{\forall}, \wp(\mathrm{Trace}(\Sigma))_{\supseteq}, \wp(\Sigma)_{\supseteq}, \gamma_M^{\forall})$. Thus, the expressive power of B(L) (in $\wp(\Sigma)$) actually can be characterized as $\mathrm{Sem}_{\wp(\Sigma)}(\mathrm{B(L)}) = \{\alpha_M^{\forall}(\llbracket\varphi\rrbracket) \mid \varphi \in \mathrm{L}\}$. Therefore, Emerson and Halpern [8] indeed define a mapping which abstracts the "trace" expressive power $\{\llbracket\varphi\rrbracket \mid \varphi \in \mathrm{L}\} \subseteq \mathrm{Trace}(\Sigma)$ of any linear time language L to a corresponding "branching time" expressive power $\{\alpha_M^{\forall}(\llbracket\varphi\rrbracket) \mid \varphi \in \mathrm{L}\} \subseteq \Sigma$.

As noted above, when L is closed under conjunction it turns out that the expressive power $\mathrm{Sem}_{\wp(\mathrm{Trace}(\Sigma))}(\mathrm{L})$ of L is an abstract domain in $\mathrm{uco}(\wp(\mathrm{Trace}(\Sigma))_{\subseteq})$ because it is intersection-closed. Moreover, since $\alpha_M^{\forall} : \wp(\mathrm{Trace}(\Sigma))_{\supseteq} \to \wp(\Sigma)_{\supseteq}$ is an abstraction map and therefore preserves least upper bounds, it turns out that $\{\alpha_M^{\forall}(\llbracket\varphi\rrbracket) \mid \varphi \in \mathrm{L}\}$ is intersection-closed as well, i.e., it is an abstract domain in $\mathrm{uco}(\wp(\Sigma)_{\subseteq})$. In our framework, this means that Emerson and Halpern define a mapping

$$\mathrm{EH}_{\forall} : \mathrm{uco}(\wp(\mathrm{Trace}(\Sigma))) \to \mathrm{uco}(\wp(\Sigma))$$

from trace abstract domains to branching state abstract domains. Thus, any trace abstraction $(\alpha, \wp(\mathrm{Trace}(\Sigma))_\supseteq, A, \gamma)$ allows us to generalize the $\mathrm{EH}_\forall$ transform from the specific branching abstraction $\alpha_M^\forall$ to the generic abstraction α: the generic transform $\mathcal{A}_\alpha : \mathrm{uco}(\wp(\mathrm{Trace}(\Sigma))) \to \mathrm{uco}(A)$ is therefore defined by $\mathcal{A}_\alpha(\mu) \overset{\mathrm{def}}{=} \{\alpha(T) \mid T \in \mu\}$. The interesting point is that $\mathcal{A}_\alpha$ gives rise to a higher-order abstract interpretation.

Theorem 6. *$\mathcal{A}_\alpha$ gives rise to a GI $(\mathcal{A}_\alpha, \mathrm{uco}(\wp(\mathrm{Trace}(\Sigma)))_\sqsupseteq, \mathrm{uco}(A)_\sqsupseteq, \mathcal{C}_\alpha)$, where $\mathcal{C}_\alpha(\rho) \overset{\mathrm{def}}{=} \{T \subseteq \mathrm{Trace}(\Sigma) \mid \alpha(T) \in \rho\}$.*

In particular, the concretization functions $\mathrm{HE}_\forall\colon\ \mathrm{uco}(\wp(\Sigma)) \to \mathrm{uco}(\wp(\mathrm{Trace}(\Sigma)))$ which is right adjoint to Emerson and Halpern's transform $\mathrm{EH}_\forall$ is defined by

$$\mathrm{HE}_\forall(\rho) = \{T \subseteq \mathrm{Trace}(\Sigma) \mid \alpha_M^\forall(T) \in \rho\}.$$

Therefore, in order to compare a linear time language L and a branching time language $\mathcal{L}$ (both closed under conjunction) Emerson and Halpern compare the abstraction $\mathrm{EH}_\forall(\mathrm{Sem}_{\wp(\mathrm{Trace})}(\mathrm{L}))$ with $\mathrm{Sem}_{\wp(\Sigma)}(\mathcal{L})$. In our approach, given any abstraction $(\alpha, \wp(\mathrm{Trace}(\Sigma))_\supseteq, A, \gamma)$, like those in Section 4, we can compare L with any language $\mathcal{L}$ whose semantic evaluation is in A by comparing $\mathcal{A}_\alpha(\mathrm{Sem}_{\wp(\mathrm{Trace}(\Sigma))})$ with $\mathrm{Sem}_A(\mathcal{L})$.

Abstractions, namely Galois connections, can be composed. As an example, our abstract interpretation-based view allows to compose Emerson and Halpern's abstraction with the partitioning abstraction:

$$\begin{tikzcd}
\mathrm{uco}(\wp(\mathrm{Trace}(\Sigma))_\subseteq)_\sqsupseteq \arrow[r, two heads, shift right, "\mathrm{EH}_\forall"'] & \mathrm{uco}(\wp(\Sigma)_\subseteq)_\sqsupseteq \arrow[l, hook', shift right, "\mathrm{HE}_\forall"'] \arrow[r, two heads, shift right, "\mathrm{par}"'] & \mathrm{Part}(\Sigma)_\succeq \arrow[l, hook', shift right, "\mathrm{pcl}"']
\end{tikzcd}$$

This is quite interesting because if L_1 and L_2 are comparable according to their expressive power they are also comparable according to their distinguishing power: this is an obvious consequence of the fact that abstraction maps are monotone. Thus, if L_1 and L_2 are incomparable in $\mathrm{Part}(\Sigma)$ they are also incomparable in $\wp(\Sigma)$. This can be helpful because comparisons in $\mathrm{Part}(\Sigma)$, i.e. based on distinguishing powers, could be easier than those in $\wp(\Sigma)$, i.e. based on expressive powers. In fact, one can compute the distinguishing power $P_\mathcal{L}$ of some state language $\mathcal{L}$ through a *partition refinement* algorithm. These can be efficient algorithms because they work by iteratively refining a current partition so that the number of iterations is always bounded by the height of the lattice $\mathrm{Part}(\Sigma)$, namely by $|\Sigma|$. Some well-known partition refinement algorithms are those by Paige and Tarjan [15] for CTL and by Groote and Vaandrager [10] for CTL-X. Moreover, there also exist partition refinement algorithms for generic state languages: see [6–Chapter 6] and [18]. Let us see an example.

Example 1. Let us consider the following two languages:

$$\mathrm{L} \ni \varphi ::= p \mid \mathrm{F}\varphi \mid \mathrm{G}\varphi \qquad\qquad \mathcal{L} \ni \psi ::= p \mid \mathrm{AF}\psi \mid \mathrm{AG}\psi$$

$\mathrm{L} \subseteq$ LTL is linear time, $\mathcal{L} \subseteq$ ACTL is branching time and we consider their standard interpretations. Notice that $\mathcal{L} = \mathrm{L}^\forall$, i.e., for any model M on a state space Σ, $\mathrm{Sem}_{\wp(\Sigma)}(\mathcal{L}) = \mathrm{Sem}_{\wp(\Sigma)}(\mathrm{L}^{\alpha_M^\forall})$. Our goal is comparing the expressive powers of $\forall\mathrm{L}$ and $\mathcal{L}$, i.e. $\mathrm{Sem}_{\wp(\Sigma)}(\alpha_M^\forall \mathrm{L})$ and $\mathrm{Sem}_{\wp(\Sigma)}(\mathcal{L})$. We show that they are incomparable by comparing their distinguishing powers $P_{\forall\mathrm{L}}$ and $P_\mathcal{L}$.

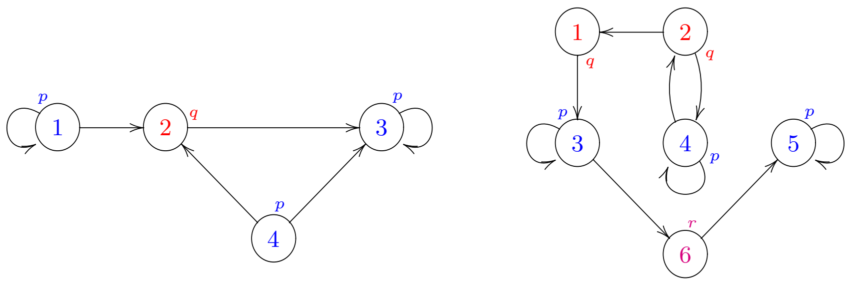

Fig. 5. Transition systems $\mathcal{T}_1$ (on the left) and $\mathcal{T}_2$ (on the right)

Let us consider the transition system $\mathcal{T}_1$ in Figure 5. Thus, the labelling for the atomic propositions p and q determines the initial partition $P = \{134, 2\}$. Let us characterize $P_{\mathcal{L}}$. We have that $[\![\mathrm{AG}p]\!] = \{3\}$, so that P is refined to $P' = \{14, 2, 3\}$. Also, $[\![\mathrm{AFAG}p]\!] = \{2, 3, 4\}$, so that P' is refined to $P'' = \{1, 2, 3, 4\}$. Hence, $P_{\mathcal{L}} = \{1, 2, 3, 4\}$. Let us now consider $\forall\mathrm{L}$. Since $\mathrm{AG}p \in \forall\mathrm{L}$, also in this case P is first refined to $P' = \{14, 2, 3\}$. It turns out that this partition can be no more refined, because:

- $[\![\mathrm{AFG}p]\!] = [\![\mathrm{AGF}p]\!] = \{1, 2, 3, 4\}$; $[\![\mathrm{AFG}q]\!] = [\![\mathrm{AGF}q]\!] = \varnothing$;
- $\mathrm{FGF} = \mathrm{GF}$ and $\mathrm{GFG} = \mathrm{FG}$.

Thus, $P_{\mathcal{L}} \prec P_{\forall\mathrm{L}}$.

Let us now consider the transition system $\mathcal{T}_2$ in Figure 5. Here, the labelling for the atomic propositions provides $P = \{12, 345, 6\}$ as initial partition. Let us characterize $P_{\mathcal{L}}$. Since $[\![\mathrm{AG}p]\!] = \{5\}$, P is refined to $P' = \{12, 34, 5, 6\}$. This partition can be no more refined because:

$$[\![\mathrm{AF}\{12\}]\!] = \{1, 2\},\ [\![\mathrm{AG}\{12\}]\!] = \varnothing; \qquad [\![\mathrm{AF}\{5\}]\!] = \{5, 6\},\ [\![\mathrm{AG}\{5\}]\!] = \{5\};$$
$$[\![\mathrm{AF}\{34\}]\!] = \{1, 2, 3, 4\},\ [\![\mathrm{AG}\{34\}]\!] = \varnothing; \qquad [\![\mathrm{AF}\{6\}]\!] = \{6\},\ [\![\mathrm{AG}\{6\}]\!] = \varnothing.$$

Thus, $P_{\mathcal{L}} = \{12, 34, 5, 6\}$. On the other hand, let us characterize $P_{\forall L}$. In this case, it is enough to notice that $[\![\mathrm{AFG}p]\!] = \{1, 3, 5, 6\}$, so that P is refined to $P' = \{1, 2, 3, 4, 5, 6\}$. Hence, $P_{\forall\mathrm{L}} = \{1, 2, 3, 4, 5, 6\}$. In this case, we have that $P_{\forall\mathrm{L}} \prec P_{\mathcal{L}}$.
Summing up, we showed that $\forall\mathrm{L}$ and $\mathcal{L}$ are incomparable in $\mathrm{Part}(\Sigma)$, i.e., they have incomparable distinguishing powers. This implies that $\forall\mathrm{L}$ and $\mathcal{L}$ have incomparable expressive powers. □

References

1. M.C. Browne, E.M. Clarke and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theor. Comp. Sci.*, 59:115-131, 1988.
2. E.M. Clarke and I.A. Draghicescu. Expressibility results for linear time and branching time logics. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, LNCS 354:428-437, 1988.
3. P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th ACM POPL*, pp. 238-252, 1977.

4. P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proc. 6th ACM POPL*, pp. 269-282, 1979.
5. P. Cousot and R. Cousot. Temporal abstract interpretation. In *Proc. 27th ACM POPL*, pp. 12-25, 2000.
6. D. Dams. *Abstract Interpretation and Partition Refinement for Model Checking*. Ph.D. Thesis, Univ. Eindhoven, 1996.
7. D. Dams. Flat fragments of CTL and CTL*: separating the expressive and distinguishing powers. *Logic J. of the IGPL*, 7(1):55-78, 1999.
8. E.A. Emerson and J.Y. Halpern. "Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic. *J. ACM*, 33(1):151-178, 1986.
9. R. Giacobazzi, F. Ranzato, and F. Scozzari. Making abstract interpretations complete. *J. ACM*, 47(2):361-416, 2000.
10. J.F. Groote and F. Vaandrager. An efficient algorithm for branching bisimulation and stuttering equivalence. In *Proc. 17th ICALP*, LNCS 443:626-638, 1990.
11. O. Kupferman and M. Vardi. Relating linear and branching model checking. In *Proc. IFIP PROCOMET*, pp. 304-326, Chapman & Hall, 1998.
12. O. Kupferman and M. Vardi. Freedom, weakness and determinism: from linear-time to branching-time. In *Proc. 13th LICS*, pp. 81-92, IEEE Press, 1998.
13. L. Lamport. Sometimes is sometimes "not never" – on the temporal logic of programs. In *Proc. 7th ACM POPL*, pp. 174-185, 1980.
14. M. Maidl. The common fragment of CTL and LTL. In *Proc. 41st FOCS*, pp. 643-652, 2000.
15. R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973-989, 1987
16. F. Ranzato. On the completeness of model checking. In *Proc. 10th ESOP*, LNCS 2028:137-154, 2001
17. F. Ranzato and F. Tapparo. Strong preservation as completeness in abstract interpretation. In *Proc. 13th ESOP*, LNCS 2986:18-32, 2004.
18. F. Ranzato and F. Tapparo. An abstract interpretation-based refinement algorithm for strong preservation. In *Proc. 11th TACAS*, LNCS 3440:140-156, 2005.
19. M. Vardi. Sometimes and not never re-revisited: on branching vs. linear time. In *Proc. 9th CONCUR*, LNCS 1466:1-17, 1998.
20. M. Vardi. Branching vs. linear time: final showdown. In *Proc. 7th TACAS*, LNCS 2031:1-22, 2001.